Making Search Relevant

Intelligent Queries. Superior Results.

AUGURI CORPORATION

2008

Authored by: Fadi Victor MICAELIAN

Table of Contents

Table of Figures

Making Search Relevant

Intelligent Queries. Superior Results.

Problems with Parametric Search

Parametric search represents the most common approach to today's querying needs, a fact which is demonstrated by the pervasiveness of SQL. Under certain circumstances and provided that query parameters are precisely crafted, this model can be quite effective. In general, however, parametric search suffers from two very significant problems.[1]

Firstly, parametric search often fails to return an appropriate number of results, instead returning far too many or too few. Being able to control the approximate size of a result set is important. A query that generates a list of e-mail recipients for a marketing campaign is ineffective if it yields only three results; a search which yields one thousand results for personal computers is equally ineffective. Inappropriately sized result sets force users to run additional queries, broadening or narrowing the search parameters. But without an intimate knowledge of the data set, crafting a query which is appropriately precise may take many iterations. Thus, poorly sized result sets are an inefficiency in time and effort.

Secondly, SQL queries often fail to deliver the most relevant results. In particular, such queries cannot distinguish between search criteria with varying degrees of importance. As an example, suppose we are searching for inexpensive hotels which are as close as possible to San Francisco International Airport. Shouldn't the results be differently prioritized for users who are price-sensitive, as opposed to users who are concerned with proximity to the airport? Parametric searches cannot make this distinction.

In today's "Information Age", solution providers face the increasingly difficult challenge of harnessing a vast universe of electronic information. In particular, query technologies have not kept up with the enormous progress of data storage technologies. This has resulted in "information overload," where large amounts of data are created and stored, but the lack of appropriate tools to extract the right information from it with a reasonable amount of effort is daunting.

How Parametric Search works

A parametric search imposes explicit search constraints on one or more parameters in the dataset, extracting only the data that meet those constraints.

SELECT ATTRIBUTE FROM DATASET WHERE CONDITION 1 AND CONDITION 2 etc..

[1] The terms *parametric search, constraint based search,* and *SQL-type search* are synonymous and will be used interchangeably throughout this paper.

Constraint-based search can be represented in n-dimensional space, where *n* is the number of attributes, by a box based on the search constraints. The search retrieves the data that meets the query constraints – that is, the data inside the "constraint box".

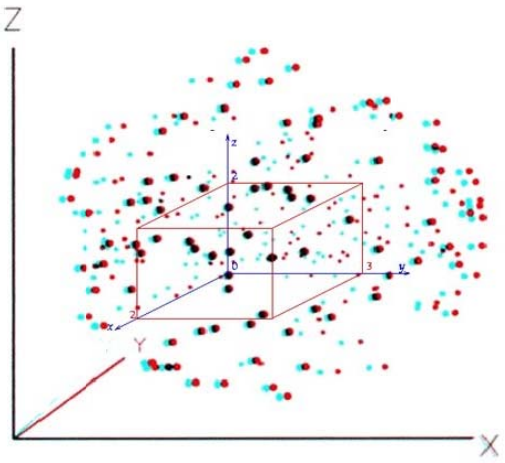

FIGURE 1: DATA THAT MEET THE QUERY CONSTRAINTS ARE INSIDE THE "CONSTRAINT BOX".

Explaining SQL Limitations with Optimization Theory

Optimization theory teaches us that ideal results are usually found close to the pareto-optimal boundaries – these boundaries are typically the intersections of curves, planes and graphs representing constraints and utility functions.

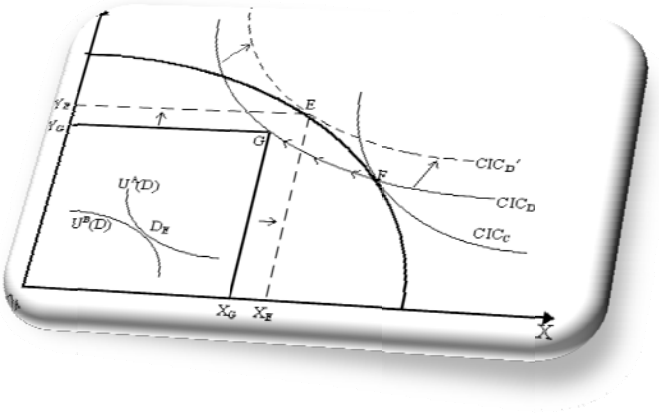

FIGURE 2: ACCORDING TO OPTIMIZATION THEORY IDEAL RESULTS LIE NEAR THE PARETO-OPTIMAL BOUNDARY

In the case of parametric search, this intersection corresponds to a corner of the "constraint box". The optimal results in a parametric search will land next to the corner of this box which represents the ideal result. This observation leads to an important realization: **Parametric searches often miss highly relevant results** that are just outside the search box (but are close to the optimal corner), despite their attractiveness to the user. This occasionally leads decision makers to make poor decisions.

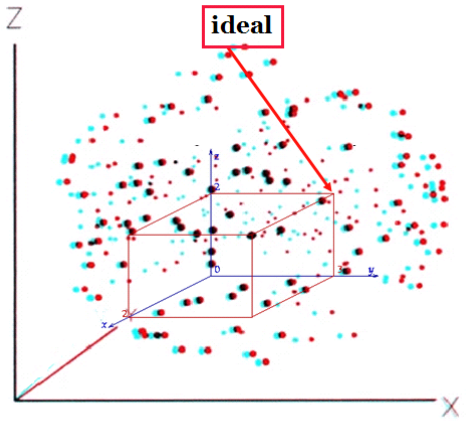

FIGURE 3: THE IDEAL RESULT COINCIDES WITH A CORNER OF THE "CONSTRAINT BOX" OF A PARAMETRIC SEARCH

In the example above, there are a number of data records very close to the ideal, but because they reside just outside the query boundaries, they will not be found by the parametric search.

Paradigm Shift

We have seen that relevant records that are only slightly on the wrong side of a query boundary will be ignored by parametric searches. An alternative solution to this dilemma would be to rank all records based on proximity to an ideal point. This would ensure that no relevant records would be missed simply because they were outside the search constraints. Furthermore, it would ensure that the right number of results could always be returned, since this number could now be set explicitly.

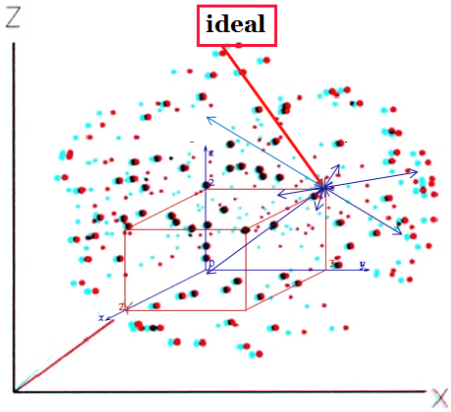

FIGURE 4: DISTANCE TO HYPOTHETICAL IDEAL REFLECT THE QUALITY OF THE RESULTS

Enriching Search with Tradeoffs

Having solved the problem of excluding relevant search results, we are positioned to address the issue of relevance. Introducing the concept of tradeoffs – the relative importance of various search criteria – provides a very elegant solution. Tradeoffs can be captured by a set of weights corresponding to each criterion. These weights result in a stretching or compressing of the corresponding axis. For example, if an Auguri user defines price as extremely important relative to performance, the axis corresponding to price will be stretched. This will effectively magnify price differences to make them more significant – data records which are more expensive will appear *even farther* from the ideal.

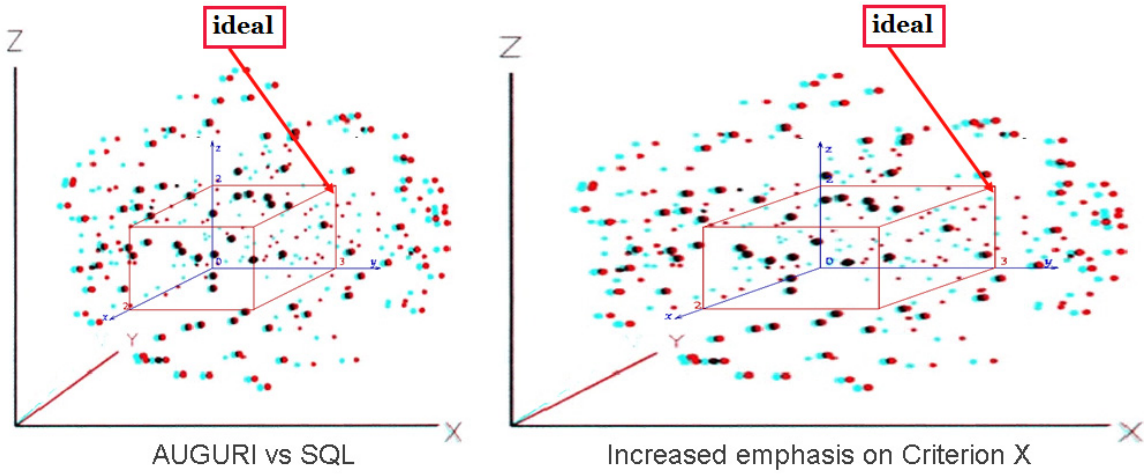

FIGURE 5: PARAMETRIC SEARCH VS. TRADEOFF SEARCH

This technique allows us to generate a mathematical notion of record "score", defined as the geometric distance from a user ideal, which is useful. The score of a record can be expressed:

$$\sqrt{\sum_{n=1}^{N} w_n (x_n - x_{in})^2}$$

where N is the number of criteria, x_{in} is the nth coordinate of the ideal, x_n is the nth coordinate of the record, and w_n is the weight of the nth criterion.

By adhering to a new model for querying which is *tradeoff-based*, Auguri searches can effectively emulate the way humans think. As an example, a tradeoff-based search model would allow users to rank search results in accordance to how closely they match the ideal situation. This is a key advantage; a parametric search for a car under $30,000 and with at least 200 horespower would yield the same results for a 'car buff" who is willing to tradeoff price (pay more) for additional horsepower and for someone who

is price sensitive. A tradeoff based query will retrieve different results for each one of these users. Auguri searches take into account the relative importance of the various criteria, yielding significantly more relevant results.

Tradeoff based search provides ability to intelligently weigh various criteria, and prioritize accordingly the result set.

Inference

Tradeoff based approach enabled Auguri to develop a particularly powerful tool - The ability to reverse engineer the query. The idea is that given a result set, the relative weights corresponding to each criteria can be derived. Put another way, given a list (partial or complete) of "best" results, it is possible to infer which criteria are most important.

Solving the inference challenge requires calculating the weights of each criterion in a set of inequalities where D is the score of a given alternative and M is the number of ranked results:

$$D_1 < D_2$$
$$D_2 < D_3$$
$$\ldots$$
$$D_m < D_{m+1} \rightarrow \sqrt{\sum_{n=1}^{N} w_n (x_{mn} - x_{in})^2} < \sqrt{\sum_{n=1}^{N} w_n (x_{(m+1)n} - x_{in})^2}$$
$$\ldots$$
$$D_{M-1} < D_M$$

This system of inequalities can be used to derive a system of weights compatible with such an ordering. We normalize the weights by having $\sum w_i = 1$, and we optimize the result by selecting the weight array that causes the least disturbance from the weights obtained from the previous session; $\lim \Delta w \rightarrow 0$.

The ability to infer a query from a result is an incredibly powerful tool which opens the door to a whole suite of capabilities. Inference can provide real time intelligence at a click of a button. Now, vast amounts of stored data can be used for analysis and true decision support.

FIGURE 6: INFERENCE IS THE ART OF REVERSE ENGINEERING A QUERY BY DETERMINING ITS SEARCH PARAMETERS

Building Blocks of Tradeoff Based Search

The objective of a search, a decision, a selection, a prioritization, a triage is to select the best solution(s) from a set of alternatives.

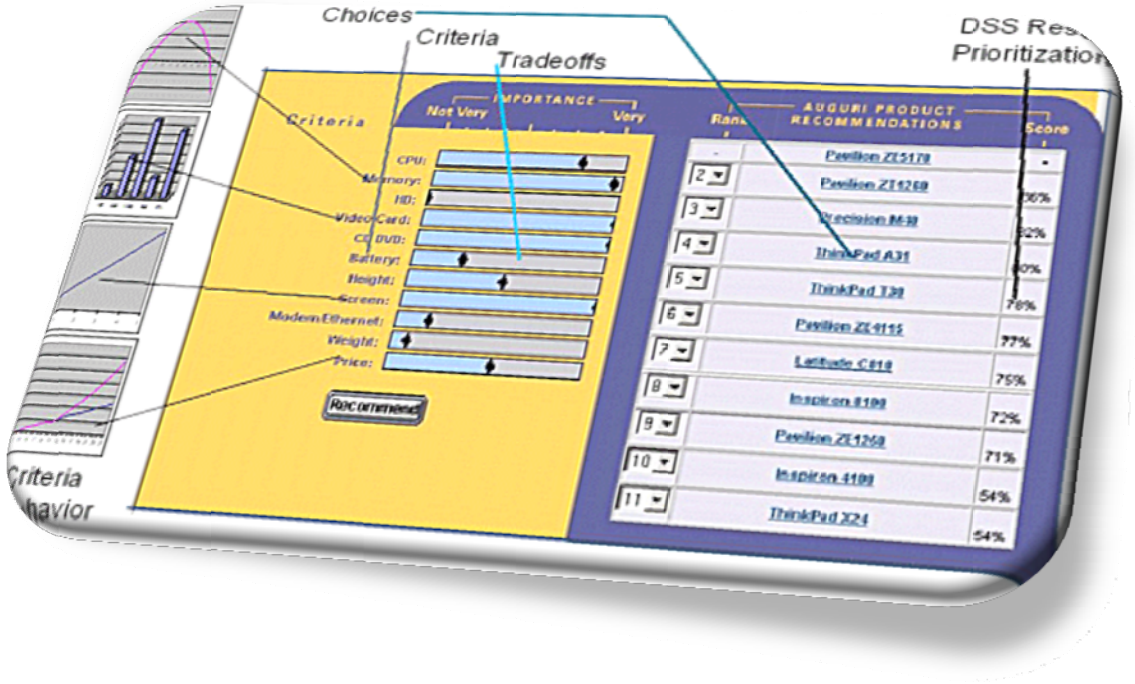

FIGURE 7: THE AUGURI SEARCH APPROACH

To be able to make a selection, the first component required is a data set which represents the set of choices. This is the database of alternatives which will be ranked according to their weighted proximity to the ideal.

In order to make a decision or conduct a selection we typically use a set of criteria.

These criteria are encapsulated by their *Criteria behavior* (shown on the left side of the Figure 7) and typically expressed by a unique function that can be thought of as a utility function, or the way we think about this specific criterion. For example a screen size is a criterion when choosing a laptop, and typically the behavior of this criterion is "the larger the better". It is important to note that these criteria are often if not always conflicting. In the same laptop selection weight is another criterion. We typically seek lighter laptops. However, the weight of the laptop increases with the size of the screen. Hence we have a conflict between these criteria. Humans have only one way to mitigate these conflicts: TRADEOFFS. Tradeoffs are a technique our brains have mastered and computers have not been able to handle until the advent of Auguri.

It is important to mention that Auguri offers an extensible library of criteria behavior. The key component of an Auguri search is *tradeoffs*, which model the importance of criteria relative to each other.

Areas of Applicability

Auguri opens the door to a number of new applications. As the technology gains exposure new areas of applicability will be discovered. Our customers have identified the following applications:

a. Enterprise Search / Selection
b. Triage
c. Decision Support
d. Business Intelligence
e. Military Intelligence
f. Prioritization
g. Procurement

Product Suite

FIGURE 8: AUGURI PRODUCT SUITE

At the heart of Auguri's products is the **AUGURI Server**. The data server performs a series of web-based services including tradeoff-based searches, inference, expert knowledge capture, profile generation, analytics and reporting services. The AUGURI Server is the kernel of the system. The Auguri server requires a database management system. Auguri provides a **decision support platform** that offers an environment for the development, deployment and implementation of web based applications that require the services performed by the AUGURI Server. The development tools include a series of published APIs as well as a web based form driven application generation solution.